

US008745441B2

(12) **United States Patent**  
**Ritz et al.**

(10) **Patent No.:** **US 8,745,441 B2**  
(45) **Date of Patent:** **Jun. 3, 2014**

(54) **PROCESSOR REPLACEMENT**

(75) Inventors: **Andrew J. Ritz**, Sammamish, WA (US);  
**Ellsworth D. Walker**, Seattle, WA (US);  
**Yimin Deng**, Sammamish, WA (US);  
**Christopher Ahna**, Puyallup, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 181 days.

(21) Appl. No.: **13/044,391**

(22) Filed: **Mar. 9, 2011**

(65) **Prior Publication Data**

US 2011/0161729 A1 Jun. 30, 2011

**Related U.S. Application Data**

(63) Continuation of application No. 11/675,290, filed on Feb. 15, 2007, now Pat. No. 7,934,121.

(60) Provisional application No. 60/866,821, filed on Nov. 21, 2006, provisional application No. 60/866,817, filed on Nov. 21, 2006, provisional application No. 60/866,815, filed on Nov. 21, 2006.

(51) **Int. Cl.**  
**G06F 11/00** (2006.01)

(52) **U.S. Cl.**  
USPC ..... **714/13**; 714/11

(58) **Field of Classification Search**  
USPC ..... 714/11, 13  
See application file for complete search history.

(56) **References Cited**

**U.S. PATENT DOCUMENTS**

5,345,590 A 9/1994 Ault et al.  
5,491,788 A \* 2/1996 Cepulis et al. .... 714/13

5,689,701 A 11/1997 Ault et al.  
5,815,731 A 9/1998 Doyle et al.  
5,913,058 A \* 6/1999 Bonola ..... 713/2  
6,199,179 B1 3/2001 Kauffman et al.  
6,226,734 B1 5/2001 Kleinsorge  
6,247,109 B1 6/2001 Kleinsorge  
6,421,679 B1 7/2002 Chang et al.  
6,502,162 B2 12/2002 Blumenau  
6,546,415 B1 4/2003 Park et al.  
6,574,748 B1 6/2003 Andress et al.

(Continued)

**FOREIGN PATENT DOCUMENTS**

CN 101542433 9/2009  
CN 101553791 10/2009

(Continued)

**OTHER PUBLICATIONS**

“Intel Hyper-Threading Technology.” Intel Corporation. Jan. 2003.\*

(Continued)

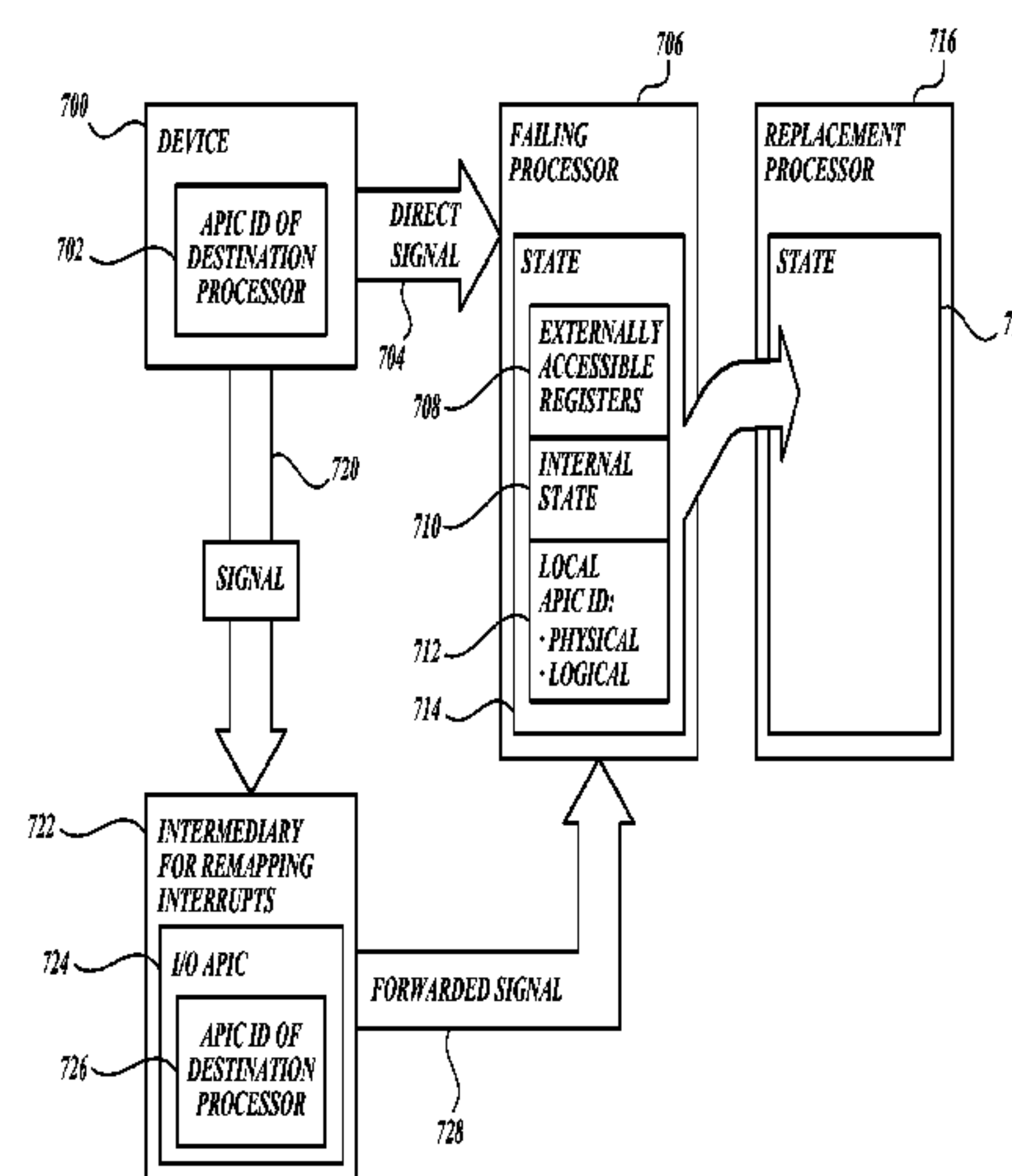
*Primary Examiner* — Philip Guyton

(74) *Attorney, Agent, or Firm* — John Jardine; Andrew Sanders; Micky Minhas

(57) **ABSTRACT**

Techniques for transparently replacing a processor, that receives interrupts in a partitioned computing device, with a replacement processor, are disclosed. In at least some embodiments, methods are discussed for directing the interrupts to an unchangeable identifier mapped to the processor's identifier and replacing the processor with the replacement processor. An intermediary, such as an I/O APIC, is used for storing the unchangeable identifier. The mapping may use logical mode delivery, physical mode delivery, or interrupt mapping.

**20 Claims, 6 Drawing Sheets**



(56)

## References Cited

## U.S. PATENT DOCUMENTS

6,598,174	B1	7/2003	Parks et al.	
6,640,278	B1	10/2003	Nolan	
6,684,230	B1	1/2004	Momoh et al.	
6,792,564	B2	9/2004	Ahrens, Jr. et al.	
6,895,586	B1	5/2005	Brasher et al.	
6,907,474	B2	6/2005	Oshins et al.	
6,986,014	B2	1/2006	Qureshi et al.	
7,007,192	B2 *	2/2006	Yamazaki	714/5.11
7,039,827	B2	5/2006	Meyer et al.	
7,051,243	B2	5/2006	Helgren et al.	
7,065,761	B2	6/2006	Foster	
7,114,064	B2	9/2006	Ramesh	
7,171,590	B2	1/2007	Kadoi	
7,275,180	B2	9/2007	Armstrong et al.	
7,343,515	B1	3/2008	Gilbertson et al.	
7,363,394	B2	4/2008	Shen et al.	
7,404,105	B2 *	7/2008	Arai	714/13
7,426,657	B2	9/2008	Zorek et al.	
7,480,911	B2 *	1/2009	Lee et al.	718/104
7,493,515	B2	2/2009	Armstrong et al.	
7,509,375	B2	3/2009	Christian	
7,543,182	B2	6/2009	Branda et al.	
7,562,262	B2	7/2009	Kondajeri	
7,877,358	B2	1/2011	Ritz et al.	
7,934,121	B2	4/2011	Ritz	
8,086,906	B2	12/2011	Ritz	
8,473,460	B2	6/2013	Ritz et al.	
8,543,871	B2	9/2013	Ritz et al.	
2002/0016892	A1	2/2002	Zalewski et al.	
2002/0053009	A1	5/2002	Selkirk et al.	
2002/0144193	A1	10/2002	Hicks	
2002/0170039	A1	11/2002	Kovacevic et al.	
2003/0037278	A1	2/2003	Olarig	
2003/0163744	A1	8/2003	Yamazaki	
2003/0212884	A1	11/2003	Lee et al.	
2004/0107383	A1	6/2004	Bouchier et al.	
2004/0193950	A1	9/2004	Gagne et al.	
2004/0221193	A1	11/2004	Armstrong et al.	
2005/0283641	A1	12/2005	Clark et al.	
2006/0010344	A1 *	1/2006	Zorek et al.	714/13
2006/0095624	A1	5/2006	Raj et al.	
2007/0061634	A1	3/2007	Marisetty	
2007/0067673	A1	3/2007	Avizienis	
2008/0010527	A1 *	1/2008	Lu	714/30
2008/0028413	A1	1/2008	Conklin et al.	
2008/0120486	A1	5/2008	Ritz et al.	
2008/0120515	A1	5/2008	Ritz et al.	
2008/0120518	A1	5/2008	Ritz et al.	
2008/0201603	A1	8/2008	Ritz et al.	
2010/0262590	A1	10/2010	Holt	
2012/0054538	A1	3/2012	Ritz	

## FOREIGN PATENT DOCUMENTS

JP	2010510592	4/2010
JP	2010510607	4/2010
KR	20090081404	7/2009
KR	20090081405	7/2009
KR	20090082242	7/2009
MX	2009004896	6/2009
WO	WO-2008064198	5/2008
WO	WO-2008064213	5/2008
WO	WO-2008073683	6/2008

## OTHER PUBLICATIONS

Nguyen et al. "Detecting Multi-Core Processor Topology in an IA-32 Platform." Intel Corporation. 2006.\*

Shanley et al. "The Unabridged Pentium 4 IA32 Processor Genealogy." Addison Wesley. Jul. 26, 2004. retrieved from Safari Tech Books Online.\*

"Advisory Action", U.S. Appl. No. 11/675,290, (Dec. 7, 2010),3 pages.

"Final Office Action", U.S. Appl. No. 11/675,290, (Jan. 27, 2010),9 pages.

"Final Office Action", U.S. Appl. No. 11/675,290, (Oct. 4, 2010),11 pages.

"International Search Report", PCT/2007/085184, (Mar. 28, 2008),2 pages.

"International Search Report", PCT/US2007/085216, (Apr. 21, 2008),2 pages.

"International Search Report", PCT/US2007/085216, (Apr. 22, 2008),2 pages.

"Non Final Office Action", U.S. Appl. No. 11/675,290, (May 24, 2010),10 pages.

"Non Final Office Action", U.S. Appl. No. 11/675,290, (Jun. 24, 2009),9 pages.

"Notice of Allowance", U.S. Appl. No. 11/675,290, (Jan. 19, 2011),7 pages.

"Written Opinion of the International Searching Authority", PCT/US2007/085180, (Apr. 21, 2008),3 pages.

"Written Opinion of the International Searching Authority", PCT/US2007/085216, (Apr. 22, 2008),4 pages.

"Written Opinion of the International Searching Authority", PCT/US2007/085184, (Mar. 28, 2008),3 pages.

Fletcher, M et al., "Realtime Reconfiguration Using an IEC 61499 Operating System", *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, San Francisco, Calif, Apr. 23-27, 2001., (Apr. 2001),7 pages.

Milojicic, D et al., "Global Memory Management for a Multi Computer System", *Proceedings of the 4th USENIX Windows Symposium*, Seattle, Wash., Aug. 3-4, 2000., (Aug. 2000),12 pages.

Moore, R.W. et al., "Knowledge-Based Grids", *Proceedings of the 18th IEEE Symposium on Mass Storage Systems*, Apr. 17-20, 2001, San Diego, Calif., 2000,pp. 29-39.

"Foreign Office Action", Chinese Application No. 200780042960.2, (Mar. 22, 2012), 6 pages.

"European Search Report", Application No. 07864653.6, (Nov. 17, 2011), 7 pages.

"Foreign Office Action", Chinese Application No. 200780042960.2, (Nov. 30, 2011), 6 pages.

"Chinese First Office Action", Application No. 200780042949.6, (Aug. 4, 2010),8 Pages.

"Chinese First Office Action", Application No. 200780042958.5, (Sep. 13, 2010),3 Pages.

"Chinese First Office Action", Application No. 200780042960.2, (Nov. 4, 2010),10 Pages.

"Chinese Notice of Grant", Application No. 2000780042960.2, (Jul. 26, 2012),4 pages.

"Chinese Notice of Grant", Application No. 200780042958.5, (Nov. 6, 2012),4 pages.

"Chinese Office Action", Application No. CN200780042958.5, Decision of Rejection,(Jun. 24, 2011), 5 Pages.

"Chinese Second Office Action", Application No. 200780042958.5, (Apr. 17, 2012), 3 pages.

"EP Communication", EP Application No. 0786453.6, (Dec. 6, 2011),1 page.

"European Search Report", Application No. 0786453.6, (Nov. 17, 2011), 7 Pages.

"Final Office Action", U.S. Appl. No. 11/675,243, (Feb. 19, 2010),12 pages.

"Final Office Action", U.S. Appl. No. 11/675,243, (Jun. 1, 2012),11 pages.

"Final Office Action", U.S. Appl. No. 11/675,261, (Oct. 5, 2010),10 pages.

"Final Office Action", U.S. Appl. No. 11/675,261, (Aug. 4, 2011), 4 pages.

"Final Office Action", U.S. Appl. No. 11/675,272, (Mar. 18, 2009), 9 Pages.

"Final Office Action", U.S. Appl. No. 13/289,776, (Dec. 6, 2012), 6 pages.

"International Search Report", Application No. PCT/US2007/085180, (Apr. 21, 2008), 2 pages.

"Issue Notification", U.S. Appl. No. 11/675,272, (Jan. 5, 2011),1 Page.



(56)

**References Cited**

## OTHER PUBLICATIONS

“Issue Notification”, U.S. Appl. No. 11/675,290, (Apr. 6, 2011), 1 Page.

“Non Final Office Action”, U.S. Appl. No. 11/675,261, (Mar. 29, 2010), 9 pages.

“Non Final Office Action”, U.S. Appl. No. 13/044,391, (Nov. 6, 2012), 11 Pages.

“Non-Final Office Action”, U.S. Appl. No. 11/675,243, (Aug. 7, 2009), 12 Pages.

“Non-Final Office Action”, U.S. Appl. No. 11/675,243, (Dec. 8, 2011), 15 Pages.

“Non-Final Office Action”, U.S. Appl. No. 11/675,261, (Mar. 4, 2011), 10 pages.

“Non-Final Office Action”, U.S. Appl. No. 11/675,272, (Aug. 7, 2009), 10 Pages.

“Non-Final Office Action”, U.S. Appl. No. 13/289,776, (Aug. 23, 2012), 11 pages.

“Notice of Allowance”, U.S. Appl. No. 11/675,243, (Feb. 26, 2013), 8 Pages.

“Notice of Allowance”, U.S. Appl. No. 11/675,261, (Aug. 24, 2011), 8 pages.

“Notice of Allowance”, U.S. Appl. No. 11/675,272, (Sep. 21, 2010), 19 Pages.

“Notice of Allowance”, U.S. Appl. No. 13/289,776, (May 13, 2013), 6 pages.

“Notice of Allowance”, U.S. Appl. No. 13/289,776, (Feb. 22, 2013), 7 pages.

“Notice of Allowance”, Application No. 70080042960.2, (Jul. 25, 2012), 6 Pages.

“Preliminary Report on Patentability”, Application No. PCT/US2007/85184, (May 26, 2009), 6 pages.

“Preliminary Report on Patentability”, Application No. PCT/US2007/85216, (May 26, 2009), 8 pages.

“Reply to Chinese Office Action”, Application No. 200780042958.5, Request for Re-Examination along with Amended Claims in the response, (Sep. 23, 2011), 14 pages.

“Reply to EP Communication”, Application No. EP0786465.6, (Jun. 15, 2012), 14 pages.

“Reply to Final Office Action”, U.S. Appl. No. 11/675,261, (Aug. 11, 2011), 13 pages.

“Reply to Final Office Action”, U.S. Appl. No. 11/675,272, (Jun. 15, 2010), 10 pages.

“Reply to Final Office Action”, U.S. Appl. No. 11/675,290, (Apr. 27, 2010), 15 pages.

“Reply to Final Office Action”, U.S. Appl. No. 11/675,290, (Nov. 24, 2010), 15 pages.

“Reply to First Chinese Office Action”, Application No. 200780042958.5, (Nov. 18, 2010), 13 Pages.

“Reply to First Chinese Office Action”, Application No. 200780042960.2, (Dec. 10, 2010), 7 Pages.

“Reply to Non-Final Office Action”, U.S. Appl. No. 11/675,261, (Jun. 29, 2010), 14 pages.

“Reply to Non-Final Office Action”, U.S. Appl. No. 11/675,272, (Nov. 3, 2009), 17 pages.

“Reply to Non-Final Office Action”, U.S. Appl. No. 11/675,290, (Aug. 20, 2010), 16 pages.

“Reply to Non-Final Office Action”, U.S. Appl. No. 12/675,290, (Oct. 26, 2009), 16 pages.

“Reply to Non-Final Office Action”, U.S. Appl. No. 13/289,776, (Nov. 27, 2012), 20 pages.

“Reply to Second Chinese Office Action”, Application No. 200780042958.5, (Jun. 15, 2012), 12 Pages.

“Reply to Second Chinese Office Action”, Application No. 200780042960.2, (Dec. 16, 2011), 2 pages.

“Reply to Third Chinese Office Action”, Application No. 200780042960.2, (May 21, 2012), 8 Pages.

“Response to Non-Final Office Action”, U.S. Appl. No. 11/675,261, (May 15, 2011), 20 pages.

“Final Office Action”, U.S. Appl. No. 11/657,243, (Feb. 19, 2010), 12 pages.

“Final Office Action”, U.S. Appl. No. 13/044,391, (Apr. 11, 2013), 17 pages.

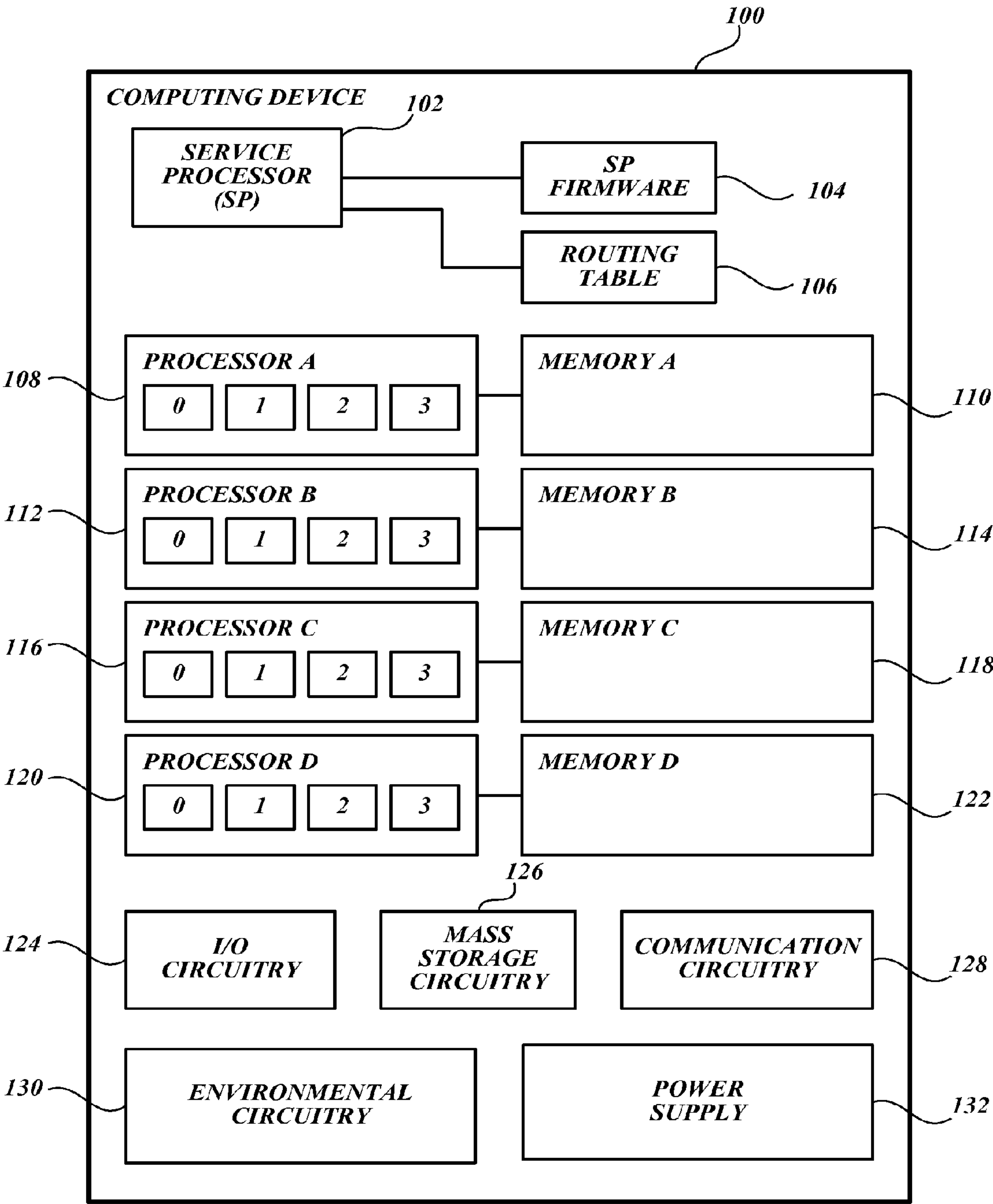
“International Search Report”, PCT Application 2007/085180, (Apr. 21, 2008), 2 pages.

“Notice of Allowance”, U.S. Appl. No. 11/675,243, (Sep. 27, 2012), 8 pages.

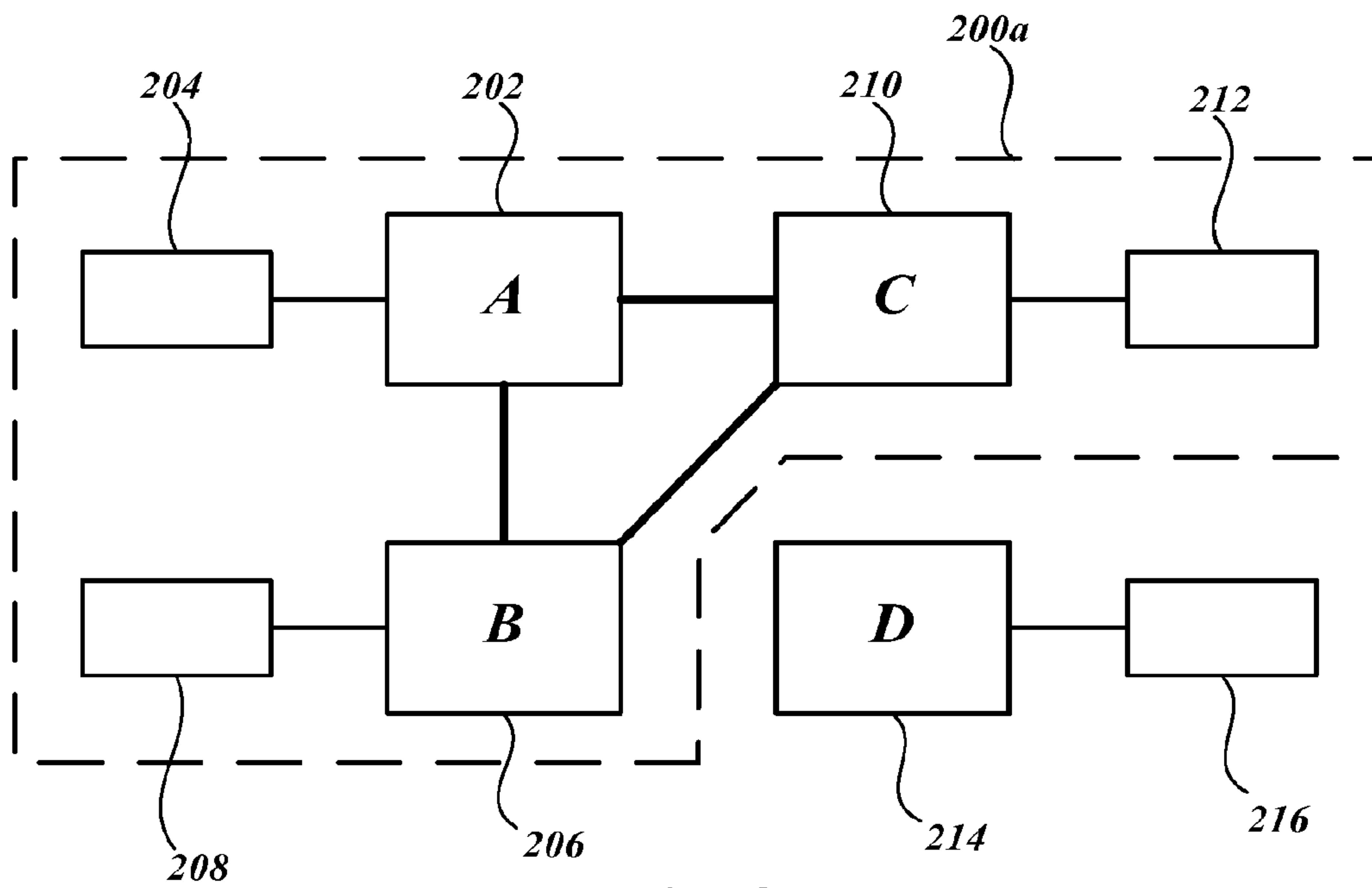
“Notice of Allowance”, U.S. Appl. No. 13/289,776, (May 13, 2013), 5 pages.

“Notice of Allowance”, Application No. CN200780042958.5, (Nov. 6, 2012), 4 pages.

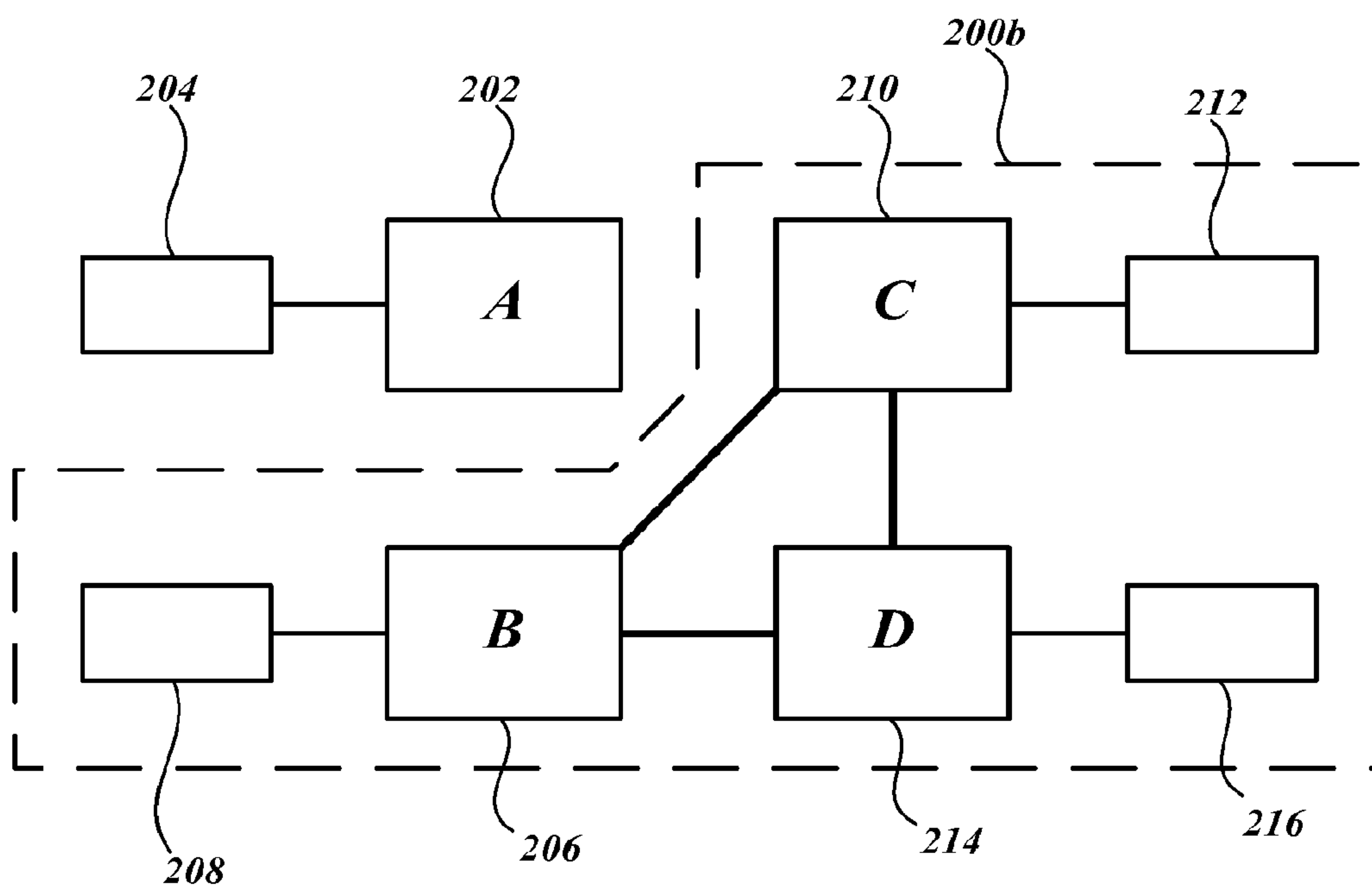
\* cited by examiner



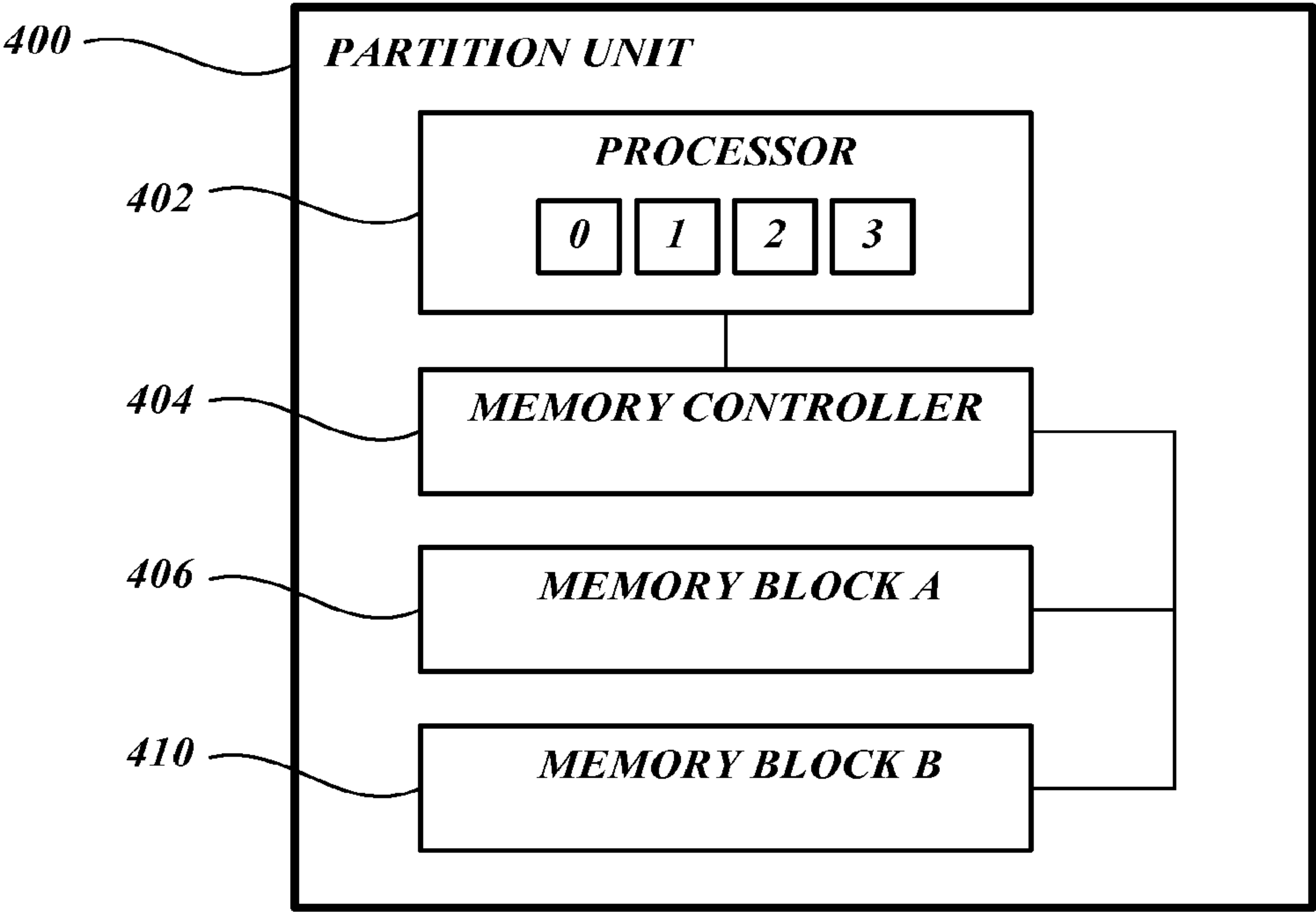
*Fig. 1.*



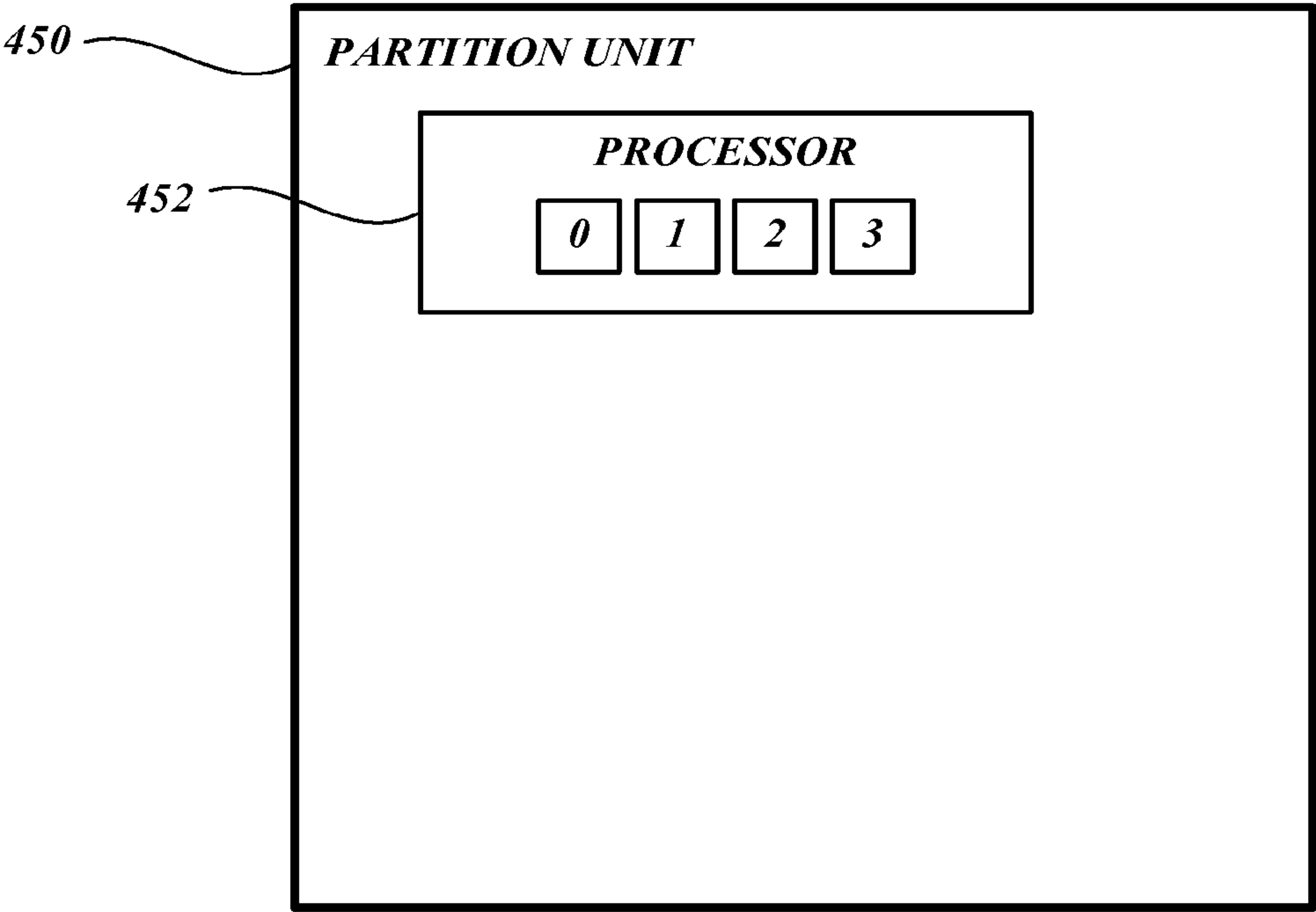
***Fig.2.***



***Fig.3.***



*Fig.4A.*



*Fig.4B.*

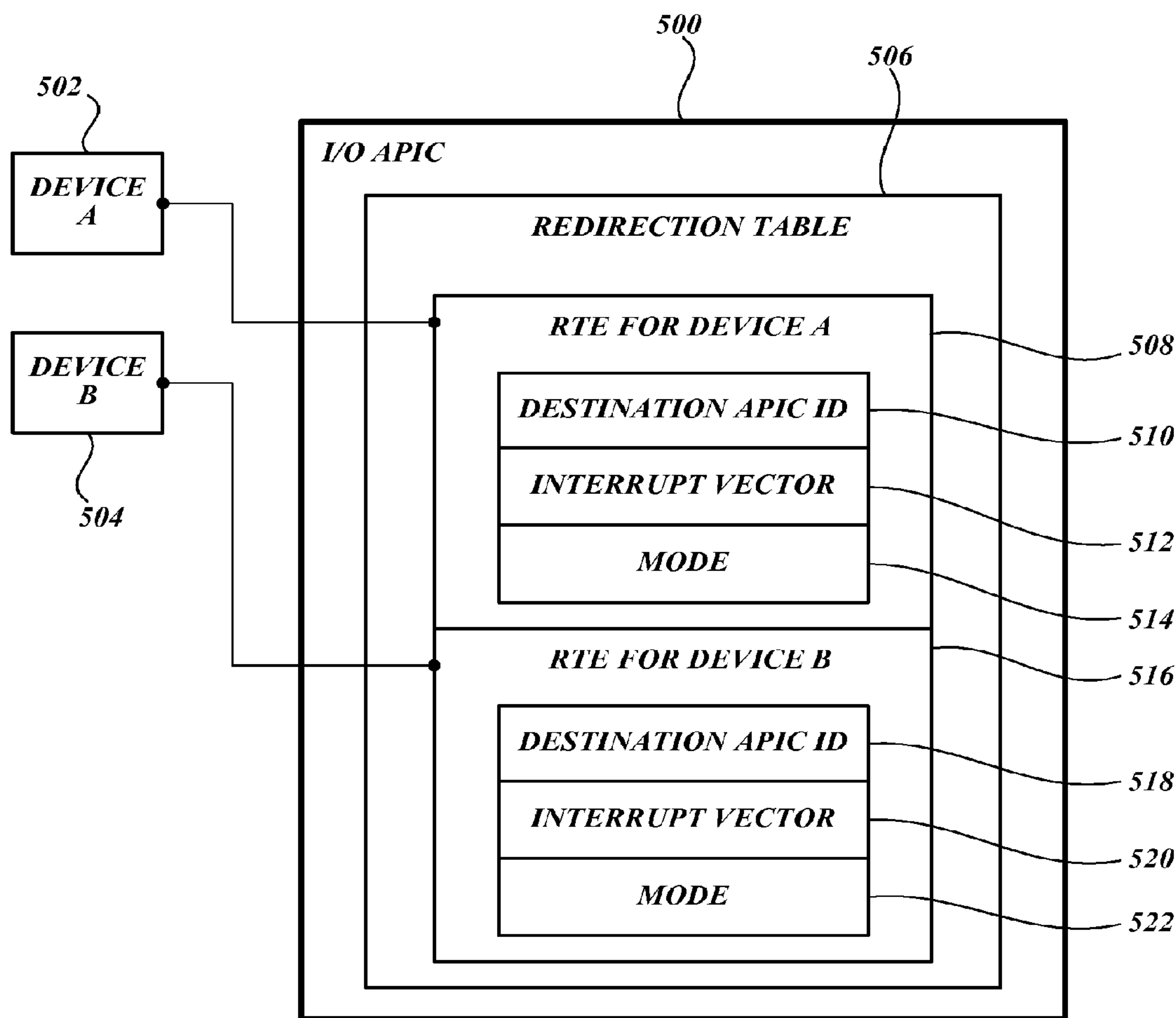
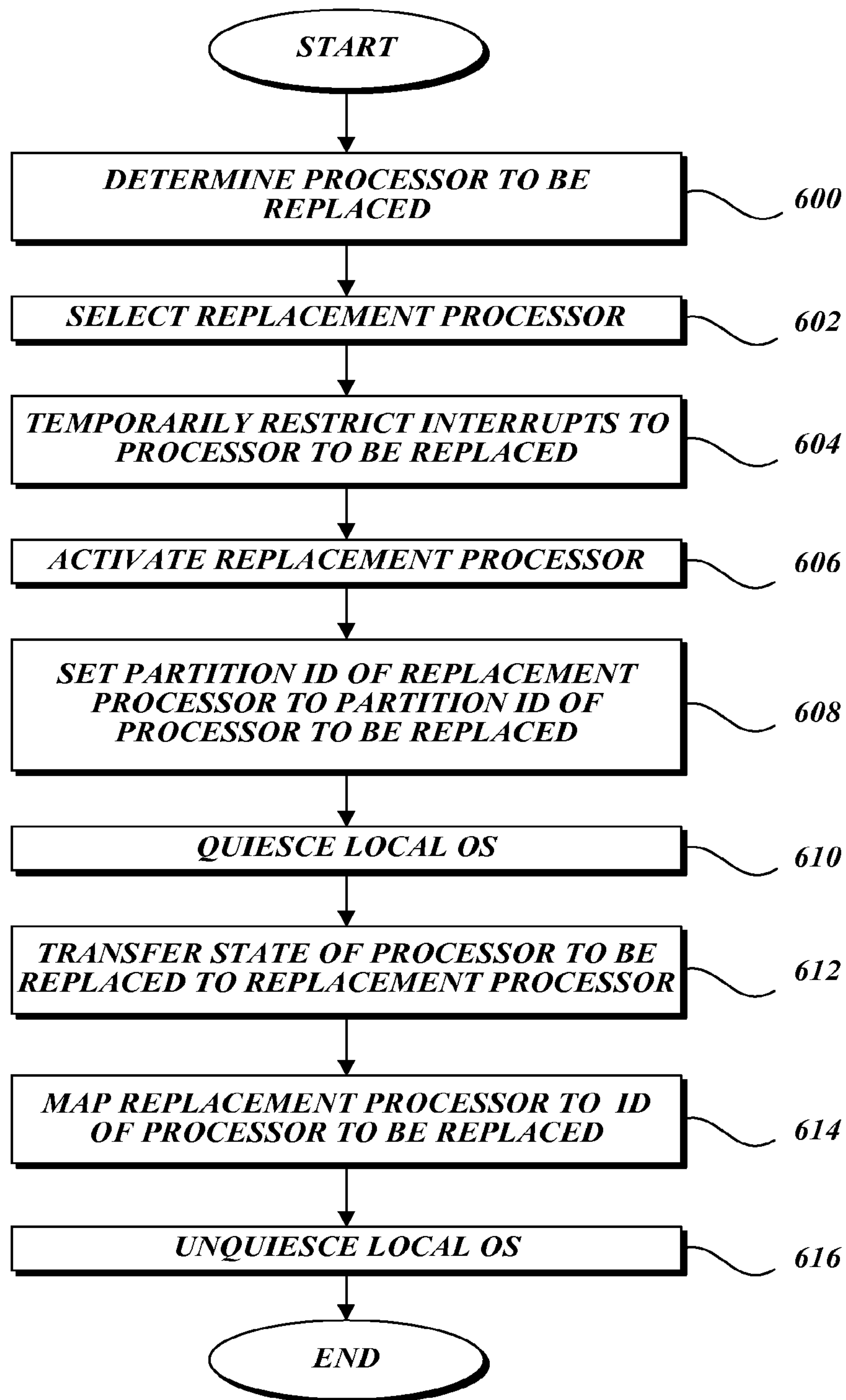
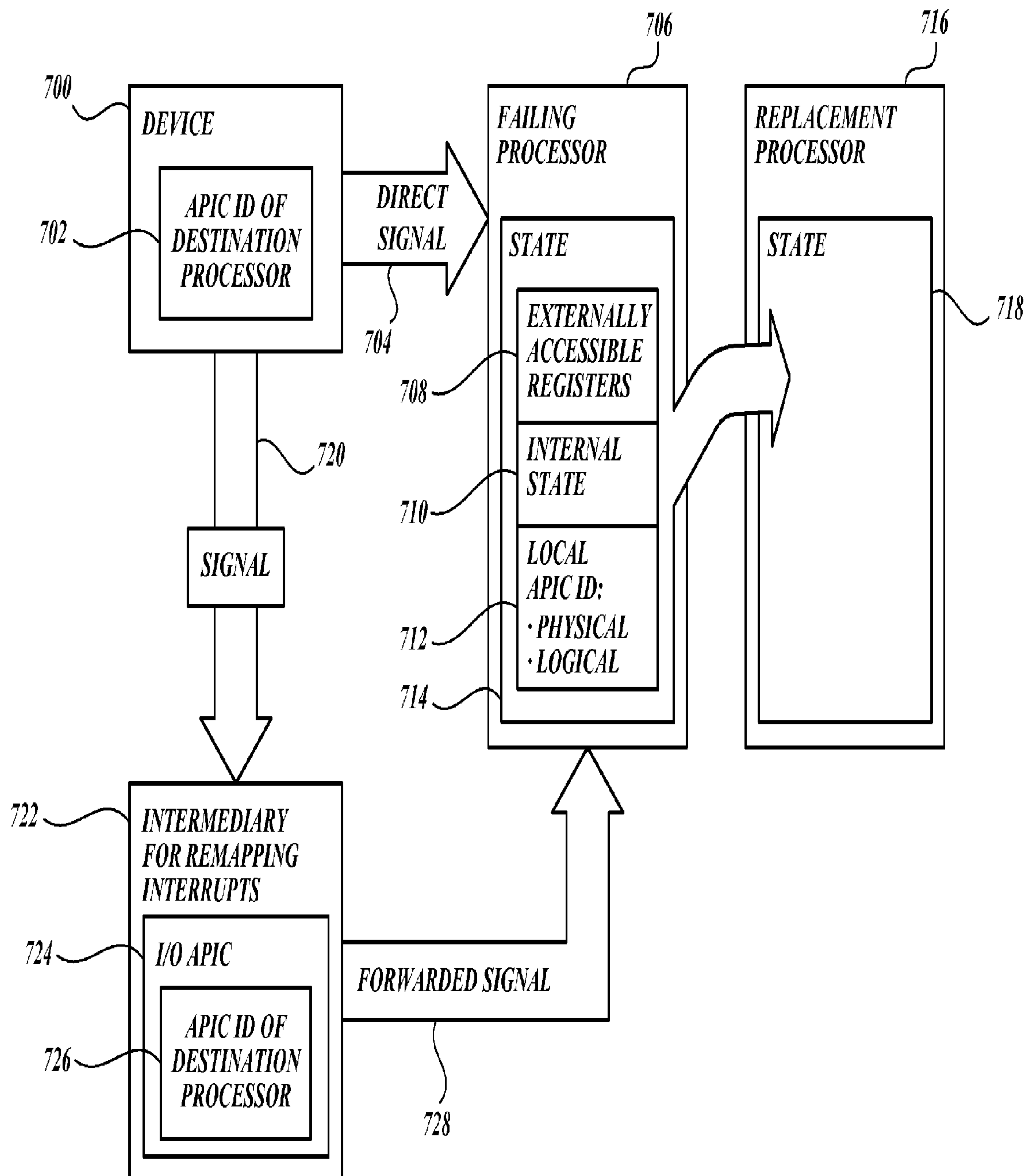


Fig.5.

**Fig. 6.**



*Fig. 7.*

**PROCESSOR REPLACEMENT****RELATED APPLICATIONS**

This application is a continuation of and claims priority to U.S. patent application Ser. No. 11/675,290, entitled "Transparent Replacement of a System Processor," filed on Feb. 15, 2007, which in turn claims priority to U.S. Pat. No. 7,877,358, entitled "Replacing System Hardware," filed on Feb. 15, 2007; and U.S. patent application Ser. No. 11/675,243, entitled "Driver Model for Replacing Core System Hardware," filed on Feb. 15, 2007; and U.S. patent application Ser. No. 11/675,261, entitled "Correlating Hardware Devices Between Local Operating System and Global Management Entity," filed on Feb. 15, 2007; and U.S. Provisional Pat. App. No. 60/866,821, entitled "Transparent Replacement of a System CPU," filed on Nov. 21, 2006; and U.S. Provisional Pat. App. No. 60/866,817, entitled "Driver Model for Replacing Core System Hardware," filed Nov. 21, 2006; and U.S. Provisional Pat. App. No. 60/866,815, entitled "Replacing System Hardware," filed Nov. 21, 2006, the disclosures of which are hereby incorporated in their entirety by reference herein.

**BACKGROUND**

A microprocessor is an electronic device capable of performing the processing and control functions for computing devices such as desktop computers, laptop computers, server computers, cell phones, laser printers, and so on. Typically, a microprocessor comprises a small plastic or ceramic package that contains and protects a small piece of semiconductor material that includes a complex integrated circuit. Leads connected to the integrated circuit are attached to pins that protrude from the package allowing the integrated circuit to be connected to other electronic devices and circuits. Microprocessors are usually plugged into or otherwise attached to a circuit board containing other electronic devices.

While a microprocessor integrated circuit typically includes only one computing unit, i.e., one processor, it is possible to include multiple processors in a microprocessor integrated circuit. The multiple processors, which are often referred to as "cores," are included in the same piece of semiconductor material and connected to the microprocessor package pins. Having multiple cores increases the computing power of the microprocessor. For example, a microprocessor with four cores can provide almost the same amount of computing power as four single-core microprocessors. Harnessing the increased computing power that multiple-core microprocessors provide allows computing functions that previously required multiple computing devices to be performed with fewer computing devices.

For example, a server implemented across 32 traditional computing devices, i.e., a 32-way server, may be implemented by eight microprocessors, each having four cores. Taking the concept one step further, if each individual core is eight times more powerful than one of the 32 computing devices, a 32-way server may be implemented by one microprocessor with four cores. Reducing the number of microprocessors reduces the cost of the server, the amount of energy required to power the server, and the amount of maintenance the server requires.

The advantages of using multiple-core microprocessors are driving a trend toward "server consolidation." Server consolidation is the process of taking multiple servers, possibly each providing a different service, and providing all of the services on one physical device, e.g., a four-core processor. While reducing costs, energy, and maintenance, consolidat-

ing servers has the effect of putting all of one's eggs into one basket. This puts a greater burden of reliability on the one physical device. If a server is implemented on many separate computing devices and a computing device fails, usually there are other computing devices that are able to take over for the failed computing device. The process of having one computing device take over for a failing computing device is referred to as "failover."

**SUMMARY**

This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This summary is not intended to identify key features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

Techniques for transparently replacing an interrupt controlled processor with a replacement processor are disclosed. Rather than directing interrupts directly to processors, interrupts are directed to an identifier mapped to a processor's identifier. Before the interrupts are directed to the replacement processor, the replacement processor's identifier is mapped to the identifier. The interrupts are directed to the identifier. The mapping of an identifier to processor's identifiers rather than directly to processors allows processors to be transparently replaced. In at least some embodiments, the identifier can be unchangeable.

A processor is replaced with a replacement processor by temporarily restricting the interrupts that are directed to the processor to be replaced; activating the replacement processor; mapping the replacement processor's identifier to the identifier; isolating the processor to be replaced; and using the mapping of the identifier to the replacement processor's identifier to direct subsequent interrupts to the replacement processor. In at least some embodiments, an intermediary, such as an I/O APIC, stores the identifier.

The mapping of the identifier to the replacement processor's identifier may be used for logical mode delivery, physical mode delivery, or interrupt mapping.

**BRIEF DESCRIPTION OF THE DRAWINGS**

The same numbers are used throughout the drawings to reference like features.

FIG. 1 is a block diagram of an example computing device capable of supporting partition unit replacement in accordance with one or more embodiments;

FIG. 2 is a block diagram of an example partition containing a plurality of partition units, one of which is unassociated, in accordance with one or more embodiments;

FIG. 3 is a block diagram of the example partition illustrated in FIG. 2 reconfigured to include the previously unassociated partition unit in accordance with one or more embodiments;

FIG. 4A is a block diagram of an example partition unit containing a processor, memory controller, and memory blocks in accordance with one or more embodiments;

FIG. 4B is a block diagram of an example partition unit containing a processor in accordance with one or more embodiments;

FIG. 5 is a block diagram of a plurality of devices connected to an example I/O APIC in accordance with one or more embodiments;

FIG. 6 is an functional flow diagram of an example process for replacing a processor in accordance with one or more embodiments; and



FIG. 7 is a block diagram of an example device interacting with an example intermediary, an example failing processor, and an example replacement processor, in accordance with one or more embodiments.

#### DETAILED DESCRIPTION

Functionally, a server is an entity on a network that provides a service. For example, a web page server provides a service that returns web pages in response to web page requests. Other example servers are an email server that returns email messages for particular users, a video server that returns video clips from a video archive, etc. Physically, a server is a stand-alone, self-contained computing device that is often connected to other servers through networking or a similar protocol. Traditionally, there is a one-to-one mapping of a functional server to a physical server. For example, an email server is implemented on one traditional physical server. If a traditional physical email server fails, the email service can be restored by replacing the failed physical email server with another physical server.

An example physical server contains a microprocessor, a memory controller, and memory blocks controlled by the memory controller. The memory controller and the memory blocks controlled by the memory controller are often referred to as a unit, i.e., a memory unit. Physical servers may also contain additional microprocessors, memory controllers, memory blocks, and other electronic devices such as interrupt processors. Hence, physical servers containing only a microprocessor and/or memory unit should be construed as examples and not limiting. As with many types of computing devices, the operation of a physical server is controlled by a software program called an operating system. A physical server executes the instructions contained in a copy of the operating system, i.e., an instance of the operating system.

Multiple-core microprocessors make it possible to implement more than one functional server on a physical server by partitioning the resources available on the physical server into individually manageable “partitions” comprising “partition units.” A partition unit comprises an electrically isolatable microprocessor, a memory unit, and/or perhaps other electronic devices, e.g., an interrupt processor. A partition comprises one or more partition units. Hence, a partition is an electrically isolatable set of partition units and electronic devices within a physical server that can run an independent instance of an operating system, i.e., a local operating system, to implement a functional server. Hereinafter, except where noted, the term “server” refers to a physical server.

In at least some embodiments, partitioning is dynamic. That is, partitioning is performed on active computing devices, i.e., computing devices that are energized and performing useful functions. Also In at least some embodiments, partitioning is transparent. That is, partition units are assigned to, or removed from, partitions with little or no impact on the services the server provides. To support dynamic, transparent partitioning, partition units are managed as whole units and not subdivided. For example, a partition unit is moved into a partition as a unit. Therefore, when a partition unit is replaced, all of the devices in the partition unit are replaced. A server that is capable of being partitioned is a partitionable server. A server system, i.e., system, comprising partitionable servers is a partitionable system. A partitionable system provides flexibility in the number and configuration of partition units and electronic devices assigned to a partition. Partitionable systems support “server consolidation.”

Server consolidation is the process of taking multiple traditional servers, possibly each providing a different service,

and providing all of the services on one partitionable server. While reducing cost, energy, and maintenance, consolidating service puts a greater burden of reliability on the partitionable server. Whereas a traditional server implemented on many separate computing devices usually has spare computing devices that are able to take over for failing computing devices, a partitionable server needs to look elsewhere for “backup” computing power. The process of having one computing device take over for a failing computing device is referred to as “failover.” Techniques have been developed for traditional server configurations to perform failover in a controlled and orderly fashion to ensure that no data is lost and no ongoing processes are interrupted during the transition from the failing computing device to the replacement computing device. In traditional server configurations the failing “computing device” was itself a server. Since servers connect to each other through a network and are not tightly tied together, work needed to be broken into small pieces and shared across the servers, i.e., packetized. This made it easy to replace a failing server since the failing server’s work packets could be re-routed. With server consolidation the overhead of the packetizing of the work is gone, but so is the ease of completely removing a server. In order to implement servers on partitionable servers that are as robust and reliable as traditional servers, similar techniques are required.

It is impractical to make partitionable servers more reliable by notifying each of the high-level software applications when a failover is required. To enable high-level software applications to respond to such a notification would require that the computer code for each application be modified to adapt to the failover. Even notifying applications would probably not be enough to provide failover without a mechanism to replace a portion of a running server, which is not usually required in traditional server configurations. Instead, it is more practical and advantageous to involve only the lowest level software in the failover and allow the upper level software, e.g., applications, to behave as though no hardware change has happened.

An implementation of an orderly, low-level, partitionable server failover involves a global management entity and one or more local operating systems. Examples of a global management entity are a service processor (SP) and a baseboard management controller (BMC). An SP is a specialized microprocessor or microcontroller that manages electronic devices attached to a circuit board or motherboard, such as memory controllers and microprocessors. A BMC is also a specialized microcontroller embedded on a motherboard. In addition to managing electronic devices, a BMC monitors the input from sensors built into a computing system to report on and/or respond to parameters such as temperature, cooling fan speeds, power mode, operating system status, etc. Other electronic devices may fulfill the role of a global management entity. Hence, the use of an SP or BMC as a global management entity should be construed as examples and not limiting.

A local operating system is an instance of an operating system that runs on one partition. Partition units, which contain logical devices that represent one or more physical devices, are assigned to a specific partition to ensure that the logical devices cannot be shared with logical devices in other partitions, ensuring that a failure will be isolated to a single partition. Such a partition unit may indicate which physical addresses are serviced by a given memory controller and, thereby, map the physical memory addresses to the memory controller and to the physical partition unit containing the memory controller. More than one partition unit may be used to boot and operate a partition. Unused or failing partition units may be electrically isolated. Electrically isolating par-



## 5

tion units is similar to removing a server from a group of traditional servers with the advantage that partition units may be dynamically reassigned to different partitions. Managing, e.g., adding or replacing, the partition units in a partitionable server allows a failover to be performed in a controlled and orderly fashion to ensure that the partitionable server is as robust and reliable as traditional servers.

An example computing device **100** for implementing a partitionable server capable of supporting partitions and partition unit addition and/or replacement is illustrated in block diagram form in FIG. 1. The example computing device **100** shown in FIG. 1 comprises a service processor (SP) **102** that is connected to a memory that stores SP firmware **104** and a routing table **106**. The computing device **100** also comprises processor A **108** connected to memory block A **110**, processor B **112** connected to memory block B **114**, processor C **116** connected to memory block C **118**, and processor D **120** connected to memory block D **122**. The processors **108**, **112**, **116**, and **120** are interrupt controlled processors, i.e., processors that are capable of responding to interrupt signals directed to the processors. Each of the processors **108**, **112**, **116**, and **120** contains four cores designated **0**, **1**, **2**, and **3**, which are capable of responding to interrupt signals.

The SP **102**, which is controlled by the SP firmware **104**, uses routing table **106** to manage the processors **108**, **112**, **116**, **120** and memory blocks **110**, **114**, **118**, and **122**. Computing device **100** also comprises I/O (input/output) circuitry **124**, mass storage circuitry **126**, communication circuitry **128**, environmental circuitry **130**, and a power supply **132**. The computing device **100** uses the I/O circuitry **124** to communicate with I/O devices. The computing device **100** uses the mass storage circuitry **126** to interact with internally and externally connected mass storage devices. The computing device **100** uses the communication circuitry **128** to communicate with external devices, usually over networks. The computing device **100** uses the environmental circuitry **130** to control environmental devices such as cooling fans, heat sensors, humidity sensors, etc. The power supply **132** powers the computing device **100**. If, for example, SP **102** is replaced by a BMC, the BMC may communicate with and control the environmental circuitry **130** and the power supply **132** more precisely.

In FIG. 1, which illustrates an example computing device for implementing a partitionable server, an example partition unit is formed by a processor A **108** and a memory block A **110**, which is connected to processor A **108**. Similarly, three more example partition units are formed by processor B **112** and memory block B **114**; processor C **116** and memory block C **118**; and processor D **120** and memory block D **122**. The four example partition units form an example partition that is managed by an SP **102**. FIGS. 2 and 3 illustrate, in diagrammatic form, an example partition similar the partition of FIG. 1 having partition units similar to the partition units formed from the processors and memory blocks shown in FIG. 1.

The replacement of partition units may be understood by comparing the block diagram shown in FIG. 2 to the block diagram shown in FIG. 3. Both of the block diagrams shown in FIG. 2 and FIG. 3 include the same four partition units. Each of the partition units comprises a processor and memory: processor A **202**, connected to memory **204**; processor B **206**, connected to memory **208**; processor C **210**, connected to memory **212**; and processor D **214**, connected to memory **216**. As discussed more fully below with respect to FIGS. 4A and 4B, partition units may include additional elements, such as a memory controller and large memory blocks or may not include memory other than the small memory associated with a processor. While the block dia-

## 6

grams in both FIG. 2 and FIG. 3 illustrate the same four partition units, the partition **200a** shown in FIG. 2 comprises a different set of partition units when compared to the partition **200b** shown in FIG. 3.

The partition **200a** illustrated in FIG. 2 comprises: processor A **202** and memory **204**; processor B **206** and memory **208**; and processor C **210** and memory **212**. In FIG. 2 the partition unit comprising processor D **214** and memory **216** is not included in partition **200a**. In contrast to the partition **200a** shown in FIG. 2, the partition **200b** shown in FIG. 3 has been changed to comprise a different set of partition units, i.e., a different set of processors and memory blocks. The partition **200b** shown in FIG. 3 comprises: processor B **206** and memory **208**; processor C **210** and memory **212**; and processor D **214** and memory **216**. In FIG. 3, the partition unit comprising the processor A **202** and memory **204** is not included in the partition **200b** whereas the partition unit comprising processor D **214** and memory **216** is included in partition **200a** shown in FIG. 2. In effect, the partition unit comprising processor D **214** and memory **216** replaces the partition unit comprising processor A **202** and memory **204**. Such a replacement would be desirable if, for example, processor A **202** and/or memory **204** were failing.

Partition **200a** and partition **200b** are in effect the same partition in that they have the same partition identifier (ID), the difference being that partition **200a** is made up of a different set of partition units than is partition **200b**. Prior to the transfer, the partition IDs of processors A, B and C were the ID of partition **200a/200b**. The partition ID of processor D was different or not set, i.e., zeroed depending on the prior status of the partition unit including processor D. Regardless of the partition ID of processor D, as explained more fully below, after the transfer the partition ID of processor D becomes the ID of partition **200a/200b**.

Replacing a partition unit involves identifying the hardware devices that are to be replaced and the replacement hardware devices. It is common for a processor, such as processor A **202**, to have an Advanced Programmable Interrupt Controller ID (APIC ID) identifying the processor. Similarly, within a partition's local operating system, a memory unit's physical address uniquely identifies the memory unit. Within a partition's local operating system, such as partition **200a**'s local operating system, a processor's APIC ID is uniquely identifies the processor. A computing device, such as computing device **100**, shown in FIG. 1, may include a plurality of partitions. Each partition in the plurality of partitions runs a local operating system having a local view of the partition. The global management entity, such as SP **102**, maintains a global namespace containing identifiers that uniquely identify each of the partitions with which the global management entity communicates.

During a partition unit replacement, such as the partition unit replacement shown in FIGS. 2 and 3 and described above, the global management entity and the local operating system of a partition communicate information concerning partition units. For example, the SP **102**, shown in FIG. 1, i.e., the global management entity, communicates with partition **200a**'s local operating system during the replacement of a partition unit. In order for the global management entity to distinguish the partition units of the same partition as well as the partition units of different partitions, the global management entity stores a unique identifier for each partition unit. The unique partition unit identifiers, i.e., partition unit IDs, enable the global management entity to access the resources, e.g., processors and memory units, of each of the partition units of each of the plurality of partitions. In contrast, a



partition's local operating system can access only the resources included in the partition.

Partition unit IDs are a combination of the partition ID and a hardware device identifier such as an APIC ID for a processor or a physical address for a memory unit. For example, to create a unique global identifier for processor C 210, processor C 210's APIC ID is combined with partition 200a's partition ID. Similarly, to create a unique global identifier for memory 212, memory 212's physical address is combined with partition 200a's partition ID.

When a partition unit is replaced, each of the hardware devices in the partition unit is replaced. For example, as shown in FIGS. 2 and 3, a first partition unit comprises processor A 202 and the memory 204 that is connected to processor A 202; and, a second partition unit comprises processor D 214 and the memory 216 that is connected to processor A 214. When the second partition unit replaces the first partition unit, processor D 214 replaces processor A 202 and the memory 216 replaces memory 204.

While a single processor and a single memory block, such as processor A 202 and memory 204, may comprise a partition unit, a partition unit may have other forms. A detailed view of an example partition unit 400 having a different form is illustrated in FIG. 4A. In FIG. 4A, as in FIG. 1, the example partition unit 400 comprises a processor 402 containing four cores: 0, 1, 2, and 3 connected to a memory controller 404 that is connected to two memory blocks—memory block A 406 and memory block B 410. The processor 402 communicates with memory controller 404, which controls memory block A 406 and memory block B 410. Another form of a partition unit is illustrated in FIG. 4B.

The partition unit 450 shown in FIG. 4B contains a processor 452 that, like the processor illustrated in FIG. 4A, includes four cores: 0, 1, 2 and 3. The partition unit shown in FIG. 4B does not include memory or a memory controller or blocks of memory. Other partition units may contain other logical devices that represent physical devices in addition to a processor, memory controller, and memory blocks and various combinations thereof well known to those skilled in the art. Hence, the partition units 400 and 450 should be construed as examples and not limiting.

A logical device in a typical partition unit may be capable of notifying the local operating system of the device's status. Alternatively, or in addition, the local operating system controlling the partition unit may use predictive analysis to assess the status of the logical device and determine if the logical device might be failing and thus, may be a candidate for replacement. While a person, such as a system administrator, might check device status as a part of regular maintenance, in at least some embodiments the hardware itself can notify the local operating system of an impending failure. In some situations, it may be desirable to upgrade a processor from one model to another model or to add processors and/or memory to a system. While a system administrator may perform such functions, in at least some embodiments such replacements and additions can be automated by using explicitly programmed instructions or by periodically timed instructions that make use of partitions, partition units, and the ability of hardware to report status.

Processes, such as the processes in a local operating system, that are running on a processor that is to be replaced must be quiesced, i.e., put into an inactive state, because if the processor is in use, the processor's state is constantly changing. If the processor's state is changing, the processor cannot be safely and reliably replaced because the processor's state cannot be safely and reliably transferred. Therefore, a pause operation is executed by the processor that is to be replaced,

e.g., a failing processor, to prevent the processor's state from changing. Those skilled in the art and others will appreciate that the process of pausing a processor to prevent the processor's state from changing is referred to as "quiescing" the processor. A system, such as the computing device 100 shown in FIG. 1, may also be quiesced by pausing or deactivating physical devices, e.g., processors A 108, B 112, etc., in the system and pausing threads of execution. Pausing or deactivating devices in the system and pausing threads of execution prevents devices and threads of execution from attempting to interrupt a processor that is to be replaced.

A partition unit, such as the partition unit 450 illustrated in FIG. 4B, may need to be replaced because the partition unit contains a failing processor, e.g., processor 452. In order to replace the failing processor, perhaps using a process such as the example process described above, in at least some embodiments the failing processor can first be identified. A way to identify a processor, failing or otherwise, is to use an Advanced Programmable Interrupt Controller identifier (APIC ID). A processor's APIC ID is stored in the processor in logical and/or physical form. A processor's logical APIC ID can be changed in the processor, usually by reprogramming a register containing the APIC ID. A processor's physical APIC ID is hardwired into the processor, i.e., fixed, and cannot be changed. An operating system, e.g., a local operating system, uses APIC IDs to, for example, schedule new threads on processors and direct interrupts to particular processors. Processors may be physical, for example processor 452 shown in FIG. 4B; or logical, for example core "0" in processor 452.

When an operating system starts, each logical processor is assigned a unique, initial APIC ID. The initial APIC ID is composed of the physical processor's ID and the logical processor's ID within the physical processor. An operating system may use initial APIC IDs to direct interrupts to particular processors. A device, e.g., a disc drive, may transmit an interrupt signal directly to the processor using a message signaled interrupt (MSI). A device interrupt may instead be routed into an intermediary software entity, i.e., an intermediary. The device generates a signal that is transmitted to the intermediary and the intermediary forwards the signal to the processor. An example traditional intermediary is an Input/Output Advanced Programmable Interrupt Controller (I/O APIC).

The block diagram in FIG. 5 illustrates certain functional parts of an example I/O APIC 500 and two example devices—device A 502 and device B 504. There may be more functional parts and more devices. Hence, the number of devices and functional parts should be construed as examples and not limiting. Example I/O APIC 500, shown in FIG. 5, contains a redirection table 506. The redirection table 506 contains redirection table entries (RTEs). The redirection table 506 comprises an RTE for device A 508 that is associated with device A 502 and an RTE for device B 516 that is associated with device B 504. The RTE for device A 508 comprises a destination APIC ID 510, an interrupt vector 512, and a mode (mode descriptor) 514. The RTE for device B 516 comprises a destination APIC ID 518, an interrupt vector 520, and a mode (mode descriptor) 522. An interrupt vector is an entry that contains, or refers, to code that handles the particular kind of interrupt. An interrupt may be conveyed over a dedicated, physical "wire," i.e., physical connection, connected to a specific entry number in the I/O APIC.

The replacement of a partition unit, such as the partition unit 450 illustrated in FIG. 4B, involves transferring the state of the failing processor to a spare processor intended to replace the failing processor. A processor's state may include,



but is not limited to, information stored in the processor's external registers, information in the processor's internal registers and circuitry, and the physical APIC ID and logical APIC ID. Not all processors are able to have their physical APIC ID transferred or changed to accept another physical APIC ID. The physical APIC ID may be hardwired inside of a processor. In such processors, the physical APIC ID of a first processor cannot be transferred to a second processor. Hence, the second processor is unable to take on the physical identity of the first processor.

There are ways to allow the physical APIC ID of a processor to be unique yet still transfer the state of the processor, e.g., a failing processor, to another processor, e.g., a replacement processor. To other entities, the two processors will appear to be identical. Thus, for example, an interrupt directed to the failing processor will instead be directed to the replacement processor.

As described above, on many processors, the APIC ID, i.e., the physical and/or logical APIC ID, may be hardwired inside of the processor preventing the identity of the processor from being transferred. Rather than relying directly on the APIC ID to identify a processor, a processor's APIC ID, i.e., physical and/or logical APIC ID, may be hidden inside of a service, making it possible to present the other parts of the system with a service for doing operations that involve the APIC ID without direct reference to the APIC ID. In discussing such a service, it is helpful to divide a system, e.g., computing device **100** into two portions—a service processor (SP) portion and a non-SP portion. The SP portion is the combination of the SP **102**, the SP firmware **104**, and the routing table **106**. The non-SP portion comprises the remaining items in the computing device **100**. To transparently replace a processor, the non-SP portion is isolated from the identity of the processor allowing the processor identifier to be remapped. Remapping processor identifiers is accomplished by a set of instructions that may be stored in the SP portion, e.g., the SP firmware **104**, or stored in other memory and pointed to by the SP firmware **104**. The set of instructions for remapping processor identifiers is referred to hereafter as the “processor remapping service”.

Rather than relying directly on a processor's APIC ID to identify the processor, the processor's APIC ID is hidden inside of the processor remapping service, making it possible to present the non-SP portion with a service for doing operations that involve APIC IDs without direct reference to the APIC IDs. The processor remapping service assigns and accepts “unchangeable” APIC IDs, i.e., APIC IDs that do not change and are used by the non-SP portion. The processor remapping service converts the APIC ID to the appropriate changeable APIC ID. Hence, during a processor replacement, whether the changeable APIC ID is physical or logical, the changeable APIC ID can be changed to refer to a replacement processor; thus, making the processor replacement transparent.

An example processor remapping service provides three processes for transparently replacing processors: logical mode delivery, physical mode delivery, and interrupt remapping. By using one of the three processes during a processor replacement, the details of which processor is being used are abstracted out and hidden from the non-SP portion.

In the logical mode delivery process, the system, e.g., computing device **100**, is configured such that the non-SP portion uses only logical APIC IDs and is not permitted to access or use physical APIC IDs. In the logical mode delivery process, the processor remapping service uses the logical APIC ID as a remapping register. A logical APIC ID of a processor can be programmed, i.e., changed, usually by low

level software. The logical APIC ID is changeable and can be easily transferred to another processor without relying on I/O APICs and MSIs. By inserting the failing processor's logical APIC ID into the replacement processor, the failing processor's logical APIC ID is mapped to the replacement processor.

If logical mode delivery is not available in a computing device and/or system, physical mode delivery may be used to transparently replace processors. Physical mode delivery involves an intermediary. An example intermediary is a redirection table in an I/O APIC. (See FIG. **5** and the foregoing description.) In physical mode delivery, an I/O APIC's redirection table, i.e., an intermediary, stores APIC IDs for processors. The APIC IDs are mapped to the processors' physical APIC IDs. A processor's APIC ID is used by the non-SP portion allowing the processor to be replaced by another processor having a different physical APIC ID without disturbing operations in the non-SP portion.

In the physical mode delivery process an I/O APIC provides an interface through which an SP interacts with the I/O APIC. The physical mode delivery process takes advantage of the already existing I/O APIC interface to program a new physical ID, i.e., the physical ID of the replacement processor, into the I/O APIC's redirection table.

In the interrupt remapping process, the processor remapping service redirects interrupts from a processor that is replaced, e.g., processor A **202**, to a replacement processor, e.g., processor D **214**. Traditionally, devices that generate interrupts, such as a disc drive controller in the mass storage circuitry **126** of computing device **100**, store the APIC ID of a processor that is intended to receive the interrupts, i.e., the destination processor. If the disc drive controller is “hardwired,” i.e., connected physically by a wire, circuit board trace, etc., to an I/O APIC, the disc drive controller sets the voltage on the wire to a level that triggers the programmed instructions in the I/O APIC's redirection table. Usually there are other devices that are connected into other entries in the I/O APIC's redirection table. Hence, there needs to be an interface at the connections of the I/O APIC to manage the arriving interrupts.

In the logical mode delivery, physical mode delivery, and interrupt remapping processes, the local operating system is quiesced and the state is transferred. In at least some embodiments, external devices are not aware of the physical APIC ID. Using logical delivery is the easiest way to accomplish that. If a device is “hardwired,” i.e., connected physically by a wire, circuit board trace, etc., to an I/O APIC, the device sets the voltage on the wire to a level that triggers the programming in the I/O APIC. It is likely that there are other devices that hook into other entries in the redirection table in the I/O APIC. The I/O APIC is, in effect, a shared state repository. Hence, there needs to be some interface at the connections of the I/O APIC to manage the shared state repository, i.e., the I/O APIC.

An example process for replacing a processor, e.g., a failing processor, is illustrated in FIG. **6**. The process begins at block **600** in which a global management entity, e.g., SP **102** shown in FIG. **1**, determines that a processor needs to be replaced. For example, the global management entity detects that the processor has produced a certain quantity and type of errors that indicate that the processor may fail and determines that the processor needs to be replaced. At block **602**, the global management entity selects a replacement processor, e.g., processor D **214** of partition **200a** shown in FIG. **2**. A replacement processor may be selected from a plurality of spare processors according to selection criteria such as speed, specialized computing features, etc. It is also possible to



## 11

select a replacement processor from a plurality of processors executing less important processes. Less important processes may be halted or moved to other processors to allow a processor to be reused as a replacement processor.

To enable the selection of replacement processors from a plurality of spare and/or other sources of processors, the global management entity may maintain a processor database that contains selection criteria values for each processor managed by the global management entity. If a processor is failing, the global management entity forms database queries that, when submitted to the database, retrieve a list of replacement processor candidates from the database. An example database query includes the identity of the failing processor and a list of relevant selection criteria and selection criteria parameters. The database, using the query, retrieves the selection criteria values of the failing processor, compares the selection criteria values of the failing processor to the selection criteria values of spare, or otherwise potentially available, processors in the processor database. The identities of processors with selection criteria values that meet the selection criteria values of the failing processor are returned, in a list, to the global management entity. The global management entity selects a replacement processor from the list, perhaps using other criteria. It is also possible for the processor database to return the identity of only one replacement processor that is then used by the global management entity. Other ways of selecting replacement processors using selection criteria are possible. Hence, using a processor database to select replacement processors should be construed as examples and not limiting.

Continuing in FIG. 6 at block 604, the global management entity temporarily restricts interrupts to the processor to be replaced to reduce the activity of the processor thus preparing the processor for replacement. At block 606, the replacement processor, perhaps selected using the selection process described above, is activated. At block 608, the global management entity sets the partition ID of the replacement processor to the partition ID of the processor to be replaced in order to “move” the replacement processor into the partition of the processor to be replaced. For example, the partition ID of processor D 214, shown in FIG. 2, is set to the partition ID of processor A 202 in order to move processor D 214 into partition 200a, the partition of processor A 202. At block 610, the local OS, i.e., local operating system, is quiesced. Those skilled in the art will appreciate that quiescing an operating system, e.g., a local operating system, is a process for reducing the activity of an operating system.

At block 612, the global management entity transfers the state of the processor to be replaced to the replacement processor. An example processor state is illustrated in FIG. 7 and described later below. At block 614, the global management entity maps the replacement processor to the identity of the processor to be replaced and updates the local OS’s state to reflect the mapping. More specifically, the replacement processor is mapped to the complete ID, i.e., the combination of the partition ID and the APIC ID, of the processor to be replaced and the local OS’s state is updated to reflect the mapping. At block 616, the local OS is unquiesced, i.e., the local OS is reactivated. After block 616, the process ends.

FIG. 7 is a block diagram showing how an example device interacts with an example intermediary to send a signal to an example failing processor, which is routed to an example replacement processor. In FIG. 7, an example device 700 contains the APIC ID of the destination processor 702. A signal 720 carries the APIC ID of destination processor 702 to an intermediary for remapping interrupts 722. The intermediary for remapping interrupts 722 contains an I/O APIC data

## 12

structure 724. The I/O APIC 724 contains the APIC ID of the destination processor 726. The device 700 may send the APIC ID of a destination processor such as failing processor 706 via a direct signal 704 or via a signal 720 through the intermediary for remapping interrupts 722, which sends a forwarded signal 728 to the destination (failing) processor 706.

The failing processor 706 has a state 714. The state 714 of the failing processor 706 is determined by the state of externally accessible registers 708, the internal state 710 of the failing processor 706, and a local APIC ID 712. The local APIC ID contains a physical address and a logical address. While the state describing a processor normally includes the state of externally accessible registers 708, the internal state 710 of the failing processor 706, and a local APIC ID 712, the state of the failing processor 706 may also include other elements or may not include one or more of the previously mentioned elements. Hence the contents of the state that describe the state of a processor should be construed as examples and not limiting. The state 714 of failing processor 706 is transferred to, or copied to, the state 718 of a replacement processor 716.

While one or more embodiments have been illustrated and described, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the disclosed embodiments.

What is claimed is:

1. A computer-implemented method comprising:

selecting a first processor associated with a computing device to be used to replace a second processor associated with the computing device;  
quiescing an operating system for the computing device responsive to a determination that the second processor is to be replaced;  
mapping the first processor to an identifier of the second processor, the mapping comprising:  
determining a logical identifier (ID) and a partition ID associated with the second processor;  
using a combination of the logical ID and the partition ID as a remapping register; and  
inserting the combination of the logical ID and the partition ID associated with the second processor into the first processor; and  
unquiescing the operating system responsive to said mapping.

2. The computer-implemented method of claim 1, wherein selecting the first processor to be used to replace the second processor is responsive to an indication of a failure of the second processor.

3. The computer-implemented method of claim 1, wherein selecting the first processor to be used to replace the second processor comprises selecting the first processor from a plurality of processors included in a database of processors, the database including selection criteria values for each of the plurality of processors.

4. The computer-implemented method of claim 1, wherein selecting the first processor to be used to replace the second processor comprises selecting the first processor from a plurality of processors based on one or more selection criteria, the selection criteria comprising a speed of the first processor or one or more specialized computing features of the first processor.

5. The computer-implemented method of claim 1, wherein the logical ID comprises an advanced programmable interrupt controller (APIC) ID.

6. The computer-implemented method of claim 1, wherein mapping the first processor to the identifier of the second processor is performed by a service processor that is associ-



## 13

ated with the computing device and is separate from the first processor and the second processor.

7. The computer-implemented method of claim 1, further comprising redirecting interrupts from the second processor to the first processor based at least in part on the mapping.

8. The method of claim 1, further comprising reconfiguring a partition associated with the second processor to include the first processor and exclude the second processor by replacing a partition ID associated with the first processor with a partition ID associated with the second processor.

9. A computer-implemented method comprising:  
determining that a first processor associated with a computing device is to be replaced with a second processor associated with the computing device;

reconfiguring a partition unit of a partition associated with the first processor to include the second processor and exclude the first processor by replacing a partition unit ID associated with the second processor with a partition unit ID associated with the first processor, the partition unit ID associated with the first processor including a combination of a partition ID and a hardware device identifier;

determining that a first identifier is mapped to a physical identifier for the first processor, the first identifier enabling communication with the first processor; and  
remapping the first identifier to a physical identifier for the second processor to enable communication to be directed to the second processor.

10. The computer-implemented method of claim 9, wherein determining that the first processor is to be replaced with the second processor comprises selecting the second processor from a plurality of processors included in a database of processors, the database including selection criteria values for each of the plurality of processors.

11. The computer-implemented method of claim 9, wherein the physical identifier for the first processor comprises an advanced programmable interrupt controller (APIC) identifier for the first processor and the physical identifier for the second processor comprises an APIC identifier for the second processor.

12. The computer-implemented method of claim 9, further comprising:

quiescing an operating system associated with the first processor prior to mapping the first identifier to the physical identifier for the second processor; and  
unquiescing the operating system after mapping the first identifier to the physical identifier for the second processor.

13. The computer-implemented method of claim 9, further comprising redirecting interrupts from the second processor to the first processor based at least in part on the mapping.

## 14

14. The method of claim 9, wherein said remapping comprises replacing the physical identifier for the first processor with the physical identifier for the second processor in a redirection table that tracks physical identifiers for a plurality of processors.

15. A device comprising:

a memory; and

a service that resides at least in part on the memory, the service being configured to enable a first processor of a plurality of processors to be replaced with a second processor of the plurality of processors, the service being further configured to perform one or more operations comprising:

determining a logical identifier and a partition ID mapped to the first processor and causing the logical identifier and the partition ID to be remapped to the second processor to enable communication to be directed to the second processor;

determining a first identifier mapped to a physical identifier for the first processor and causing the first identifier to be remapped to a physical identifier for the second processor such that communication is directed to the second processor; and

causing a partition associated with the first processor to be reconfigured to include the second processor by replacing a partition ID associated with the second processor with the logical identifier and the partition ID associated with the first processor.

16. The device of claim 15, wherein the logical identifier comprises an advanced programmable interrupt controller (APIC) identifier.

17. The device of claim 15, wherein the service is further configured to enable the first processor to be replaced with the second processor responsive to a detection by the service of a failure of the first processor.

18. The device of claim 15, wherein the service is further configured to be implemented by a service processor that is separate from the plurality of processors.

19. The device of claim 15, wherein the one or more operations further comprise causing an operating system of the device to be put into an inactive state during at least one of the logical identifier and the partition ID being remapped to the second processor, or the first identifier being remapped to the physical identifier for the second processor.

20. The device of claim 15, the one or more operations further comprising causing the first identifier to be remapped to the physical identifier for the second processor by replacing the physical identifier for the first processor with the physical identifier for the second processor in a redirection table that tracks physical identifiers for the plurality of processors.

\* \* \* \* \*